DTIC FILE COPY

(2)

AD-A223 624

RT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: None ~~UNCLASSIFIED~~

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

DTIC ELECTE JUN 27 1990 S D

4. PERFORMING ORGANIZATION REPORT NUMBER(S): 1536920

5. MONITORING ORGANIZATION REPORT NUMBER(S): ~~AFOSR-86-0189-4~~ AFOSR-TR- 90 0671

6a. NAME OF PERFORMING ORGANIZATION: Department of Electrical and Computer Engineering

6b. OFFICE SYMBOL (If applicable):

7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research

6c. ADDRESS (City, State and ZIP Code): Campus Box 425, Boulder, CO 80309

7b. ADDRESS (City, State and ZIP Code): Building 410, Bolling Air Force Base, D.C. 20332

8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR

8b. OFFICE SYMBOL (If applicable): NE

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-86-0189

8c. ADDRESS (City, State and ZIP Code): Bldg 410, Bolling AFB, DC 20332-6448

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2305	B1	

11. TITLE (Include Security Classification): Optical Signal Computing (U)

12. PERSONAL AUTHOR(S): Cathey, Wade Thomas; Schmidt, Rodney A.; and Moddel, Garret

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 86-5-1 TO 89-12-31	89-12-31	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Optical computing; artificial intelligence; symbolic logic; optical processing; spatial light modulators

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Architectures for optical symbolic computing were designed, ~~[1-4]~~, devices were designed and built that were specifically for the architectures ~~[5-14]~~, and test circuits for some of the logic elements were designed, constructed, and operated. The research led to novel architectures for optical symbolic computing. Devices were developed that are suitable for optical 2-D memory and logic. These devices are pixilated photo-addressed spatial light modulators (SLMs) with a three-terminal arrangement so that the threshold can be adjusted. ~~5~~ Spinoff non-pixilated devices are useful as high frame rate, high-resolution SLMs that can be used for many optical signal processing applications ~~[6-14]~~

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION: ~~UNCLASSIFIED~~

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
~~W. Thomas Cathey~~ Craig	~~(303) 492-7366~~ 202-767-4931	NE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

90 06 26 017

SECURITY CLASSIFICATION OF THIS PAGE

PAPERS RESULTING FROM AFOSR-86-0189

1 R.A. Schmidt, "System architecture of an optical reasoning system," Optical Engineering 28, pp. 410-416 (April 1989).

2. R.A. Schmidt and W.T. Cathey, "Optical implementations of mathematical resolution," Applied Optics 26, pp. 1852-1858 (15 May 1987).

3. R.A. Schmidt and W.T. Cathey, "Optical representations for symbolic logic," Optical Engineering 27, pp. 475-481 (June 1988).

4. W.T. Cathey and R.A. Schmidt, "Low loss polarization based optical logic gates," Congress of the International Commission for Optics, SPIE Proceedings, Vol 813 (24 August 1987).

5. R.A. Rice, G. Moddel, I. Abdulhalim, and C.M. Walker, "A three-terminal spatial light modulator optically addressed by an a-Si:H photosensor," Proc. of ICAST (1st International Conference on Amorphous Semiconductor Technology) Asheville, NC, 21-15 August 1989. Also to appear in a special volume of Journal of Non-Crystalline Solids.

6. W. Li, R.A. Rice, and G. Moddel, "Hydrogenated amorphous silicon photosensor for optically addressed high speed spatial light modulator," to appear in IEEE Trans. Electron. Devices (December 1989).

7. G. Moddel, K.M. Johnson, W Li, R.A. Rice, L.A. Pagano-Stauffer, and M.A. Handschy, "High-speed binary optically addressed spatial light modulator," Appl. Phys. Lett. 55. pp.537-539 (7 August 1989).

8. R.A. Rice, W. Li, and G. Moddel, "High-speed optically-addressed spatial light modulator for optical computing," Topical Meeting on Optical Computing, 27 February-1 March 1989, Salt Lake City, UT. Paper MF1-1 in Optical Computing, 1989 Technical Digest Series, Vol. 9, Optical Society of America, pp. 64-66.

9. C.T. Kuo, R.A. Rice, W. Li, and G. Moddel, "Hydrogenated amorphous silicon photosensor for optically-addressed high-speed spatial light modulators," Conference Record of the International Topical Conference on Hydrogenated Amorphous Silicon Devices and Technology, 21-23 November 1988, pp. 259-262.

10. G. Moddel, K.M. Johnson. M.A. Handschy, "Photoaddressing of high speed liquid crystal spatial light modulators," SPIE vol. 754 (1987).

11. M.A. Handschy, K.M. Johnson, W.T. Cathey, and L. A. Pagano-Stauffer, "Polarization-Based optical parallel logic gate utilizing ferroelectric liquid crystals," Optics Letters Vol. 12, August 1987.

12. R.A. Schmidt, and G. Moddel, "Memory systems for optical computing," AIAA Computers in Aerospace VI Conference (AIAA, Washington, D.C. 1987), pp. 201-206.

13. W. Li, C.T. Kuo, G. Moddel, and K.M. Johnson, "Design and performance of high-speed photoaddressed spatial light modulators," Proc. SPIE 936 (1988).

14. G. Moddel, C.T. Kuo, K.M. Johnson, and W. Li, "Optical addressing of high-speed spatial light modulators with a-Si:H," Amophous Silicon Technology, (Materials Research Society, Pittsburgh 1988).

SUMMARY OF RESEARCH
AFOSR-86-0189

This report describes the results of the work on optical symbolic computing that was performed on grant AFOSR-86-0189 during a three year program. The detail of the research is in the appended papers. This section of the report is to summarize the motivation for the work, the reasons for the approach taken, and the results. First, a rationale is given for the use of optics in A.I., and the approach taken is described. The results of the research are then given.

MOTIVATION

Several areas of artificial intelligence, notably knowledge-based systems, pattern recognition and neural networks require so much computer power that their usefulness as problem-solving techniques are limited by present computer hardware. Our ability to address the computing power limitations of conventional processors is limited by electronic problems in the distribution of data and control signals. These problems are not present in optical computing systems. However, conventional serial approaches to A.I. problems do not map gracefully onto optical processors. A new approach is needed.

Our work was to develop an optical data representation; design, simulate and implement basic optical operations; design, simulate and implement optical storage and data handling modules; design, simulate and implement data-dependent control; and simulate and implement a prototype optical inference engine. We have developed the notation, algorithm, and logic for optical resolution; designed the resolution system using medium scale optical building blocks; generated optical and functional simulations; designed, simulated and built prototype components; and performed a functional simulation of the system.

RATIONALE FOR OPTICS

This project investigated the use of optics in implementing a particular approach to knowledge-based systems - the approach known as mathematical resolution. This combination of method and technology must compete with other technologies, for example VLSI and with other methods, such as back chaining, (i.e. PROLOG), and production rules. Our purpose here is to compare these approaches and show why mathematical resolution using optics is a viable and attractive approach.

The underlying thread in all knowledge-based systems is that they contain a large amount of unstructured information. By "unstructured", we mean that the relationship between different pieces of information is not strongly hierarchal. Thus the process of computation involves a relatively unselective examination of combinations of data to find sequences of information within the data which satisfy the requirements of logic and solve the problem at hand.

If the data space is large, many subsequences of information exist within the system which can be independently discovered. This offers the possibility of increasing the speed of a knowledge-based system by the use of parallelism, either electronic or optical. However,, the search for subsequences must have overall coordination so that all meaningful possibilities are examined, but that the same possibilities are not redundantly examined.

There is a tradeoff between the size of the data space and the complexity of the algorithms required to manipulate it. In the simplest notation, the propositional calculus, each entity in the problem space must be separately described in the data. For example, "IF FELIX IS A CAT, THEN FELIX HAS WHISKERS." If there were many cats in the problem, each would require its own statement about whiskers. In a more complex notation, the first order propositional calculus, more general statements may be made. For the example given, "IF x IS A CAT, THEN x HAS WHISKERS." This reduces the size of the initial data space, but requires considerably more complex processing to determine the applicability of a statement to a given situation. If the number of entities (e.g. cats) in a problem is small, the increase in data space size may be compensated for by the reduction in the complexity of processing, and by increased opportunities for parallelism.

Even with the data notation chosen, there remain other significant choices about how to process the data. The three primary methods are production rules (forward chaining), such as used in Mycin, backchaining, and mathematical resolution. Backchaining is the method used by the programming language PROLOG, and resolution is typically used for proving theorems in mathematical axiom systems. Each has advantages and disadvantages.

A major advantage of forward chaining is that it is relatively easy to incorporate probabilities into the data. Thus, if a given situation could have multiple causes, the relative likelihood of the alternatives can be computed. In order for this to be practical, however, the depth of the inference process and the number of alternatives must be small, otherwise the computation gets out of hand. This is the primary disadvantage of forward chaining. The process is not strongly goal-directed, and in a complex situation will (given sufficient time) deduce **everything** about the situation, in an upredictable order. If the user's interest is focussed on only one facet of the problem, there is no way to obtain directly information about only that facet.

Backchaining starts with a completely or partially specified conclusion, and reasons backwards to check its compatibility with the data. It can be used to answer questions by using conclusions from the first order predicate calculus. For example, the conclusions "x HAS WHISKERS" would result in the substitution "x=FELIX" if the fact "FELIX IS A CAT" were added to the statement in the earlier illustration. In most cases, backchaining is faster than production rules when the desired outcome can be partially specified, because the processing is more narrowly focussed. More general questions, such as "tell me the significance of the present situation", cannot be answered by backchaining systems.

Both backward and forward (production) systems have a strong directional basis. The rules only work one way. But from a logical point of view, the rules are actually bidirectional. The rule "IF x IS A CAT, THEN x HAS WHISKERS" also implies that if x does not have whiskers, it is not a cat. Mathematical resolution is a relative of backchaining which uses a different notation (clause form) which captures this bidirectionality. In addition, clause form notation is simpler from an implementation standpoint than the more general formula used in forward and back chaining. Normally, mathematical resolution is used for proof by contradiction. In this form, a hypothetical conclusion is negated, and then the data space consisting of the negated hypothesis and and the known facts is examined for inconsistency. If one exists, the original hypothesis must have been true. Resolution can also be used in the forward direction to merely generate more facts from an initial data space. In this mode, it is as undirected as production rules in forward systems.

Because of its generality and simplicity of its implementation, we have chosen to study the implementation of mathematical resolution as a parallel inference mechanism. In order to keep system complexity under control at this stage of the research, we have restricted our efforts to the propositional calculus.

Mathematical resolution has a large amount of inherent parallelism, limited only by the size of the data space. Two principal parallel operations are resolvent generation and duplicate detection. Resolvent generation is the process of combining existing clauses to form new ones. This process can be carried out independently and in parallel for every pair of clauses in the data base. Of course, this requires an adequate number of processing elements and parallel access to the data. Once new clauses are generated, it is necessary to determine whether or not they already exist in the data base. Failure to delete duplicate clauses not only causes the data space to grow at an unnecessarily high rate, but it also may cause the computation to be nonterminating. The search for existing clauses which match a new one may be done in parallel by "broadcasting" the new clause into the data base and waiting for an appropriate response. Again, parallel access to the data and a sufficient number of computing elements are required.

The computing operations required for resolution are relatively simple boolean algebra and a small amount of counting. The necessary logic could be fabricated in either VLSI or planar integrated optics. Certainly the present state of the art in fabrications favors VLSI. However, the parallel data access requirements of the problem cannot be met in VLSI. A system performing parallel logic on a 1000 x 1000 grid would require 1,000,000 bits of information per clock cycle arriving in parallel. Pinout and crosstalk limitations in VLSI would require this information to be partially serialized for loading. Even if the VLSI were faster than the equivalent optics (and there is reason to believe that the speed of optics can be made comparable), the time required for serial data handling would limit the speed of a VLSI implementation.

RESEARCH METHOD

The approach was to first develop an optical data representation for knowledge and then to construct simulation software for all optical components to validate system behavior and reduce implementation risk. The next step was to design, simulate, and implement optical computation modules for the basic operations required for inference; optical storage and data handling modules required for data-dependent computation; and the data-dependent control required for inference. The ultimate goal was to simulate and implement a prototype optical inference engine.

As was discussed in the Rationale for Optics, the A.I. technique chosen was reasoning by resolution. In reasoning by resolution, one must begin with a set of information about the problem. Then, one takes the fact of interest and states its opposite. If the fact to be proven is true, the total set including the negation now contradicts itself. The object of resolution is to make this contradiction explicit. The technique used is to convert each fact to a binary bit string (a clause) encoding the data and systematically generate new clauses by combining existing clauses. Then, new clauses are retained which contain exactly one item of information which was true in one parent and false in the other. This item is omitted in the combination and all other new clauses are discarded. The remaining new clauses are checked against all existing clauses and duplicates discarded. New clauses are generated until either an empty clause is generated or no new clauses can be formed. If the empty clause is produced, the original fact was true, otherwise it was false.

The optical inference system was designed around medium scale optical components. Major components studied and developed were:

- Parallel read - parallel write memory storage frames which allow parallel access to entire matrices of stored data.
- Switchable prism multiplexors and shifters which allow spatial switching of bit vector and matrix data.
- Optical pushdown stacks which are a combination of the memory frames and the prism shifters. These allow vector-at-a-time access to data matrices.
- Resolver (clause combiner) using medium scale boolean logic to produce a matrix of clauses from an old matrix and a new clause.
- Duplicate detector to compare a new clause against the existing clause base in parallel to rapidly reject duplicates.

RESULTS

A system was designed and simulated. Several components were constructed and tested. Photo addressed spatial light modulators (SLM) that are suitable for use as memory planes were designed and built. The work is continuing on an unfunded basis, and the demonstration of the use of the photo addressed SLM is being done as a master's thesis project. A pixilated SLM was designed and an initial device fabricated. That research is being continued at a low level using other funds. The unpixilated photo addressed device has better than 100:1 contrast ratio, 70 lines/mm resolution, and a strong 5 KHz frame rate.

Data representations were selected and analyzed, the system was designed and functional simulation was performed. Phenomenological simulations for the key elements of the system were performed. Two versions of a prototype gate have been built and tested and a Proof-of-Concept 2:1 multiplexor was built and tested. A dual bit photoaddressed FLC memory element has been built and tested.

The system was functionally simulated, varying several system design parameters. In approximate order of importance, these parameters are:

1) Parallel versus serial frame compaction. In parallel compaction, valid resolvents are immediately squeezed to the top of the frame. In serial compaction, the whole frame must be shifted and checked. This has a major impact on performance.

2) Generation storage strategy. In separated storage, data generated in each pass through the system is kept separate from previous data. This takes no more storage, but reduces computing. In compacted checking, frames are added to until full.

3) Extent of duplicate checking. In complete checking, duplicates within the same pass (generation) are detected. In partial checking, only older duplicates are found.

4) Extent of duplicate checking. In parallel checking, the entire data store is checked in one operation. In serial checking, one frame of the data store is checked at a time. This affects the performance if the faster options of the preceding choices are selected.

Table 1, gives the result of that simulation The particular problem chosen is given in Appendix I.

The data representation chosen was the positional dibit notation for clauses (the basic unit of knowledge in mathematical resolution). [1986 SPIE Conference on Optical Computing] This is illustrated in Figure 1.

The fundamental data element in a clause is the literal, which can occur in asserted form (X) or negated form (X'), or simply be omitted. Each row in the notation corresponds to a clause. Each pair of columns in the notation corresponds to a literal, with the bit pair 01 used for assertion, 10 used for negation, and 00 for omission. Each bit is encoded in polarized light as shown in Figure 1.

problem size: 15 rules, 19 clauses, 31 literals

ALGORITHM: 'SET-OF-SUPPORT - Serial timing 35,609 cycles

FRAME COMPACTION STRATEGY:

DUPLICATE DETECTION STRATEGY ---	← Serial →		← Parallel →	
	separated	compacted	separated	compacted
complete check - parallel	62,290 (175%)	41,289 (116%)	676 (1.9%)	1,682 (4.7%)
newest generation not checked - parallel	136,911 (384%)	20,020 (56%)	1,182 (3.3%)	1,128 (3.2%)
complete check - frame-by-frame	62,865 (176%)	41,533 (116%)	1,251 (3.5%)	1,926 (5.4%)
newest generation not checked- frame-by-frame	138,295 (388%)	20,452 (57%)	2,566 (7.2%)	1,560 (4.4%)

GENERATION STORAGE STRATEGY

TABLE I SIMULATION RESULTS

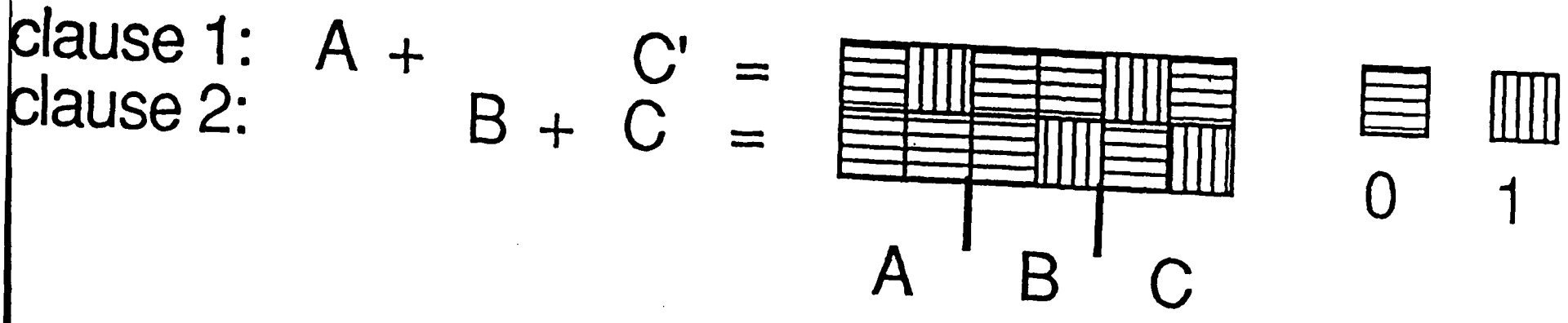

FIGURE 1, DIBIT NOTATION

The data handling and storage used a functional and data flow design of a data compaction system and was described in Applied Optics, May 1987. A review of plausible optical data storage techniques was given at the 1987 Computers in Aerospace Conference. Because data-dependent control is needed, a review of control techniques for mathematical resolution was performed. The set-of-support strategy was selected as offering the best balance between efficiency and complexity of optical implementation.

The optical resolution system is shown in Figure 2. In this figure, clauses are stored in the "old generation" push down stack. Frame-at-a-time multiplexors are used to route current information to the resolvent formation logic, which systematically combines a single clause with an existing frame to produce a frame of potential new resolvents. These are checked for validity and for duplication of existing data, and useful new resolvents are placed in the new generation. As resolution proceeds, new generation data is added to the old generations to allow the process to proceed. Figure 3 shows the operation of duplicate detection.

Ferroelectric liquid crystals are key to many of the components of the system. In one application, the fast low-loss polarization rotation characteristics are used and in others, their potential as a high-speed switch is employed. Figure 4 illustrates both of these uses.

Figure 5 shows how the switching of the index of refraction can be employed to fabricate a position shifter. This concept has been demonstrated and a shifter was fabricated. A shifter of this sort could be used to effect compaction of sparse matrices into dense matrices as shown in Figure 6.

A logic gate has been designed that uses polarization logic. It is illustrated in Figure 7, where the possible inputs are the four combinations of polarized light shown at station "A". The outputs at station "D" implement the "or" or "and" boolean logic function depending on whether logic 1 is chosen as horizontal or vertical polarization.

A simulation and an experiment were performed as shown in Figure 8. Robust operation demonstrated.

A parallel storage device with photoaddressed spatial light modulators (SLMs) has been designed and construction of the photoaddressed SLM has been completed. This device has been shown to have better than 100:1 contrast ratio, 50 lines/mm and 5 KHz frame rates. The storage device is shown in Figure 9.

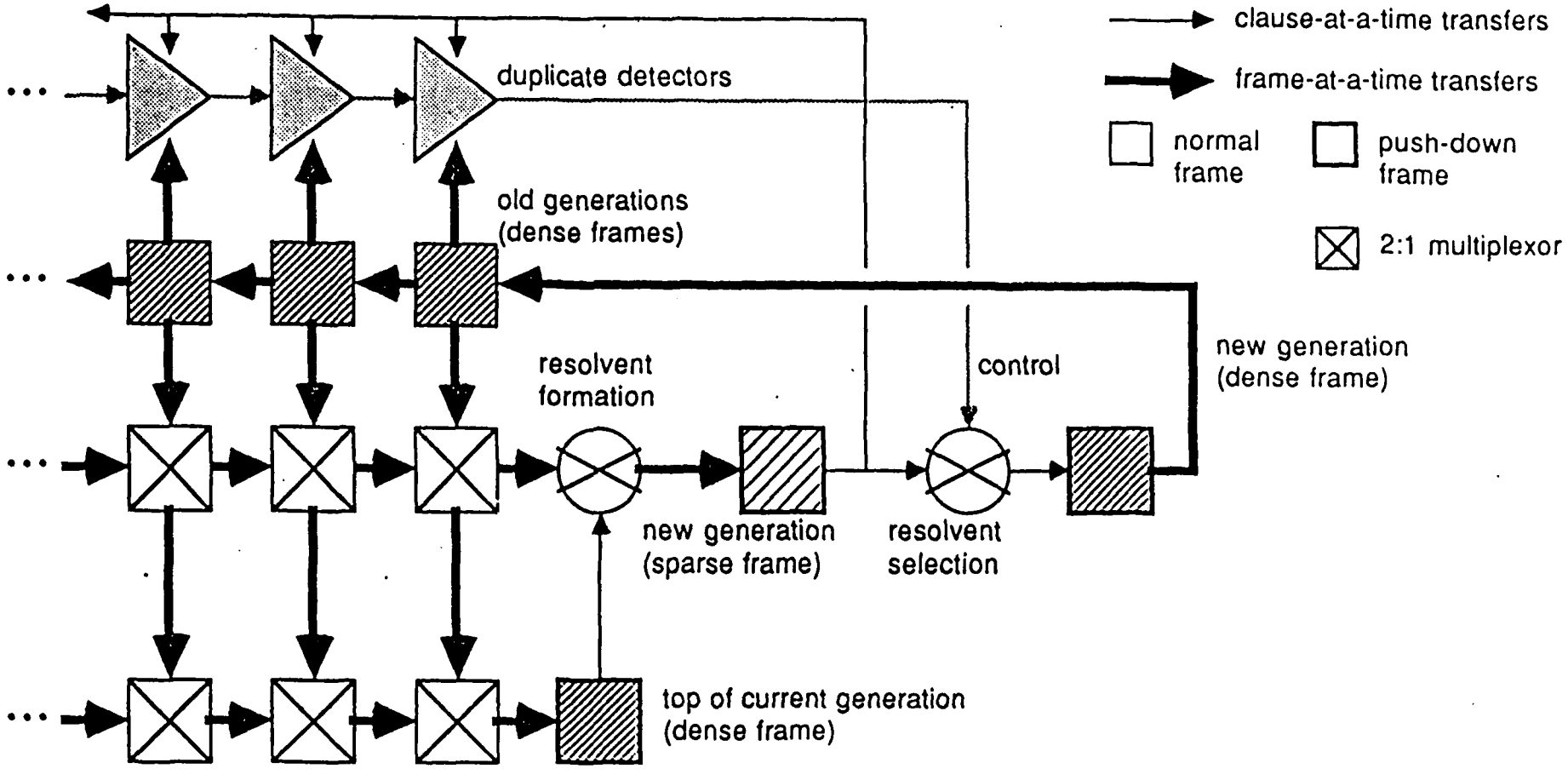

FIGURE 2, OPTICAL RESOLUTION SYSTEM

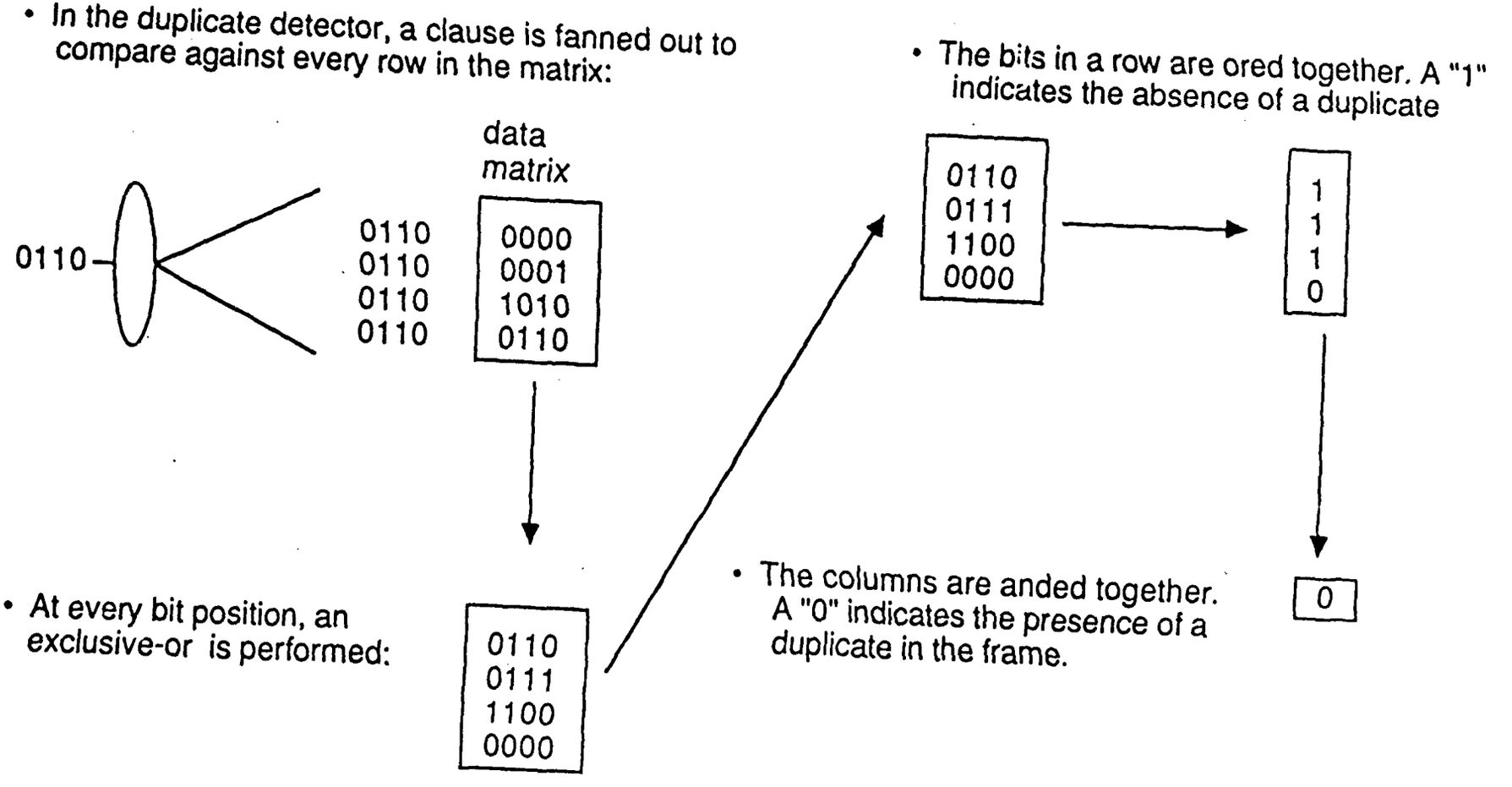

FIGURE 3, DUPLICATE DETECTION

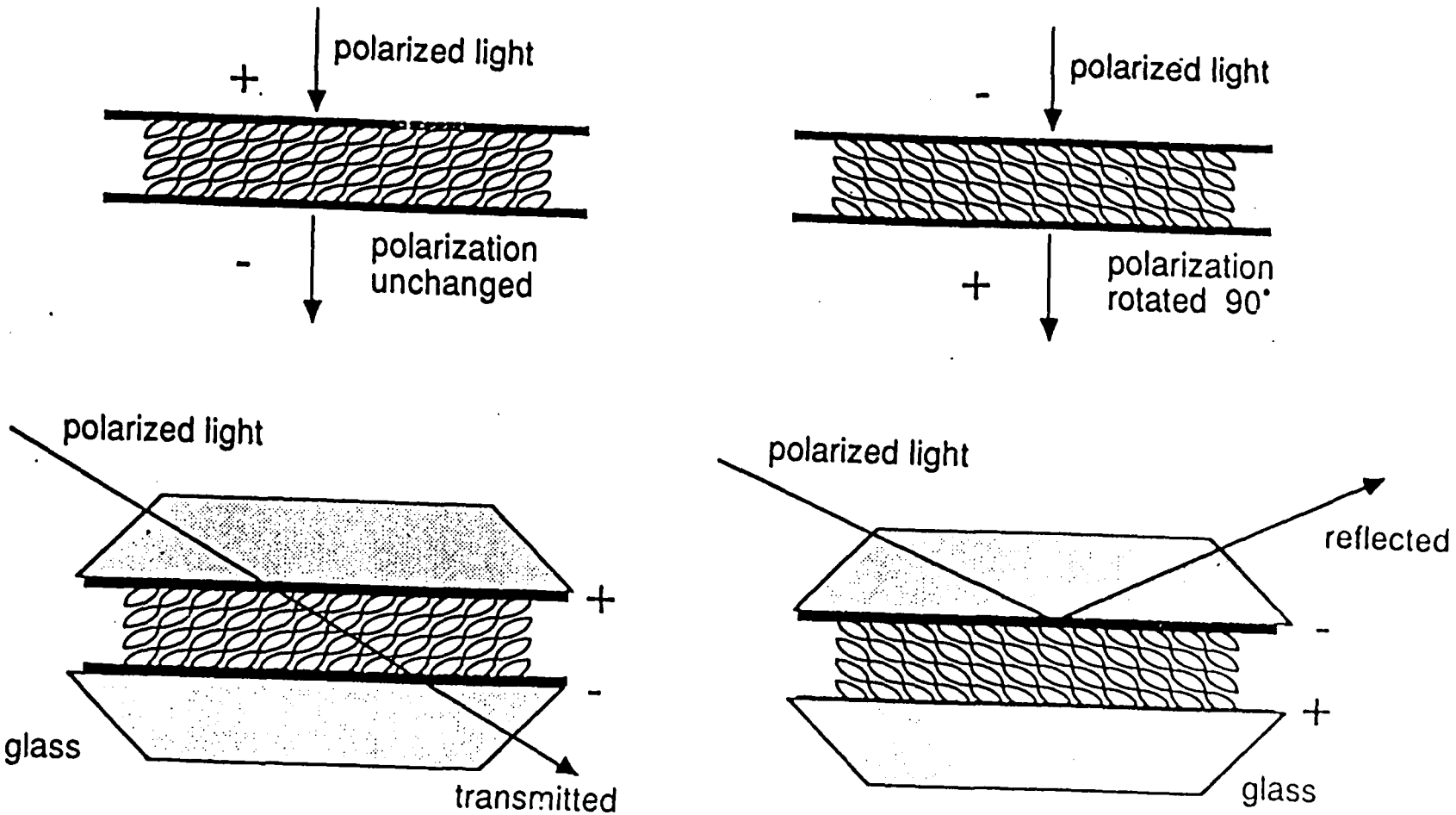

FIGURE 4, FERROELECTRIC LIQUID CRYSTALS

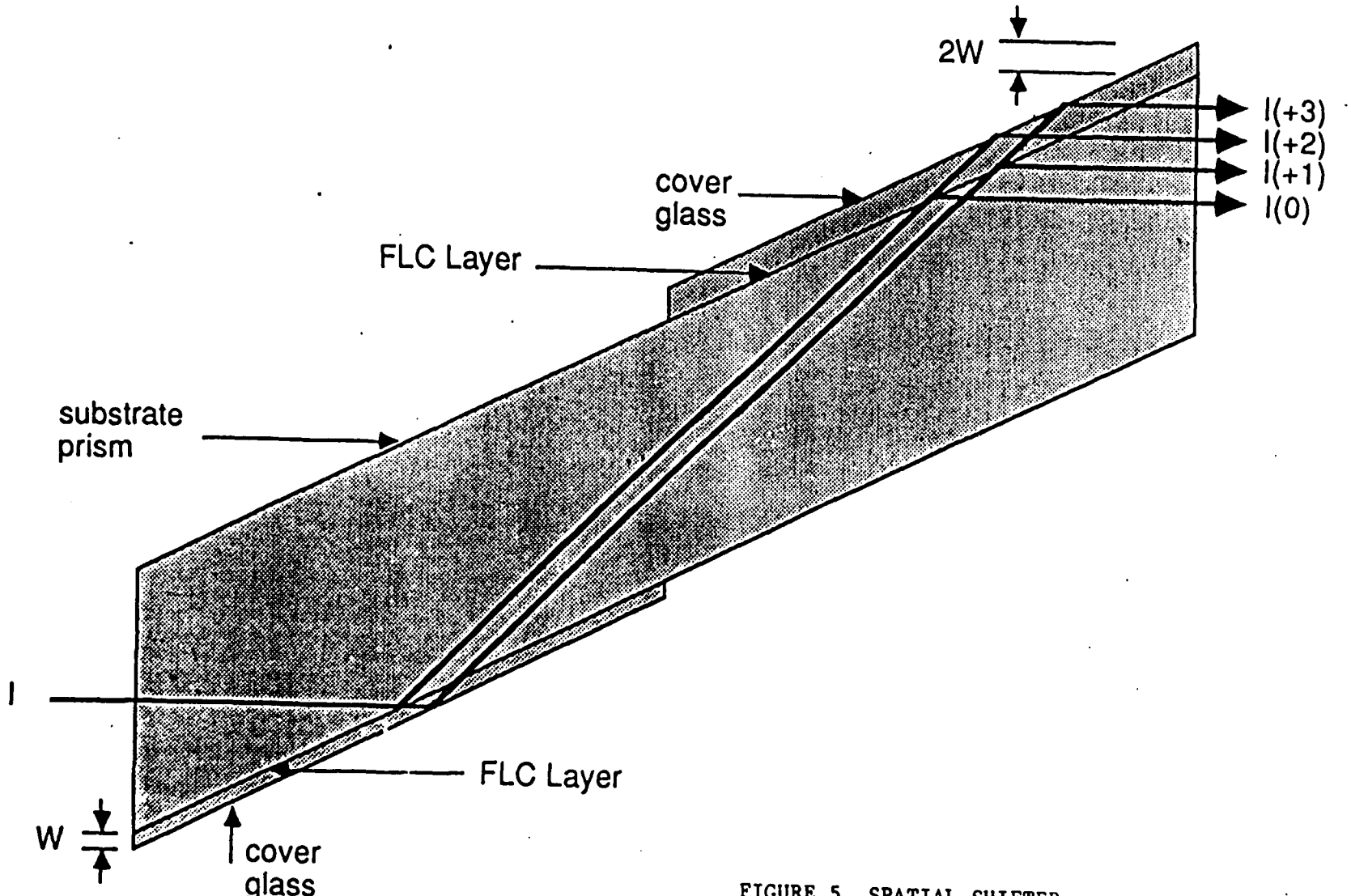

FIGURE 5, SPATIAL SHIFTER

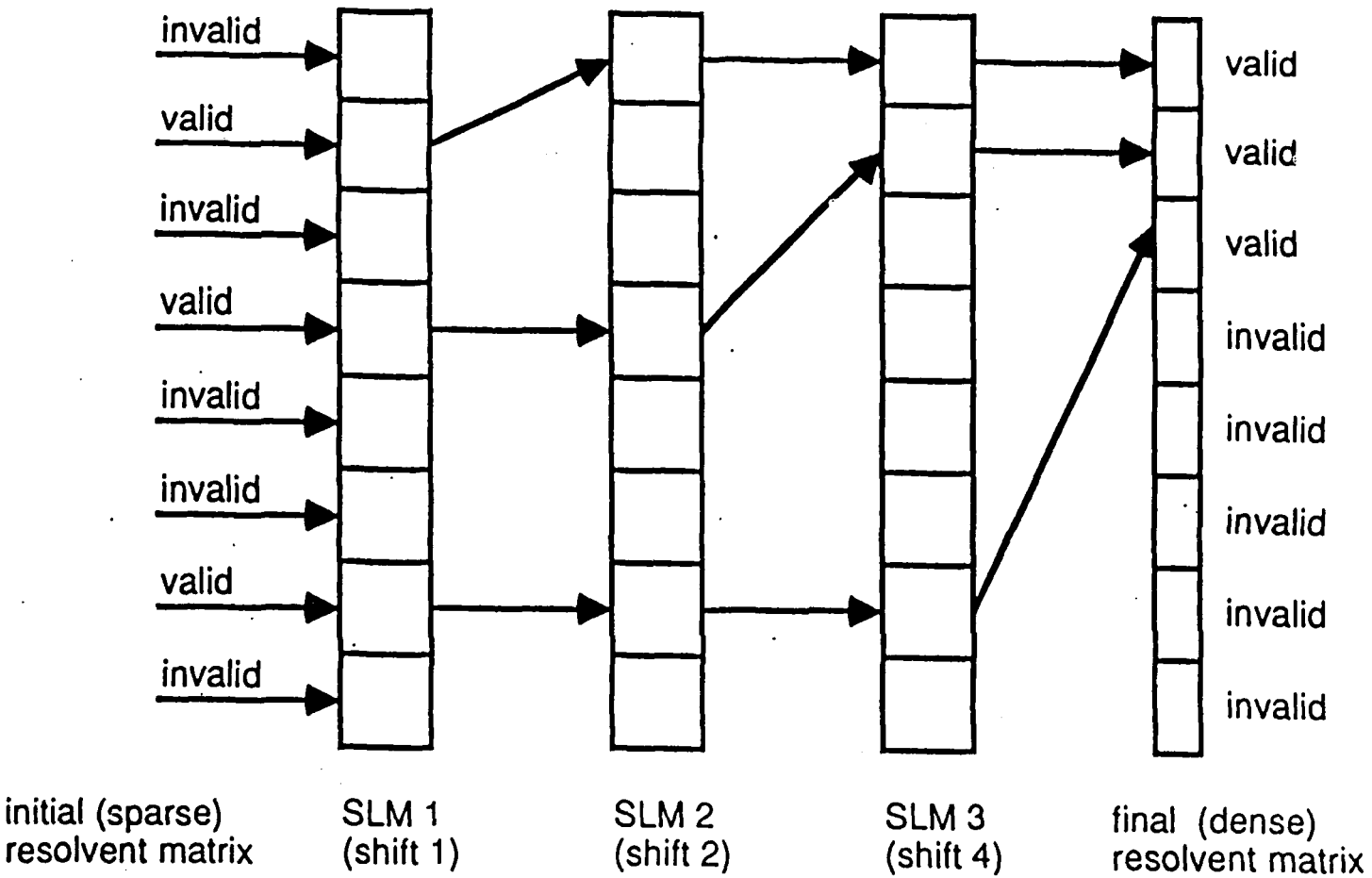

FIGURE 6, COMPACTION LOGIC

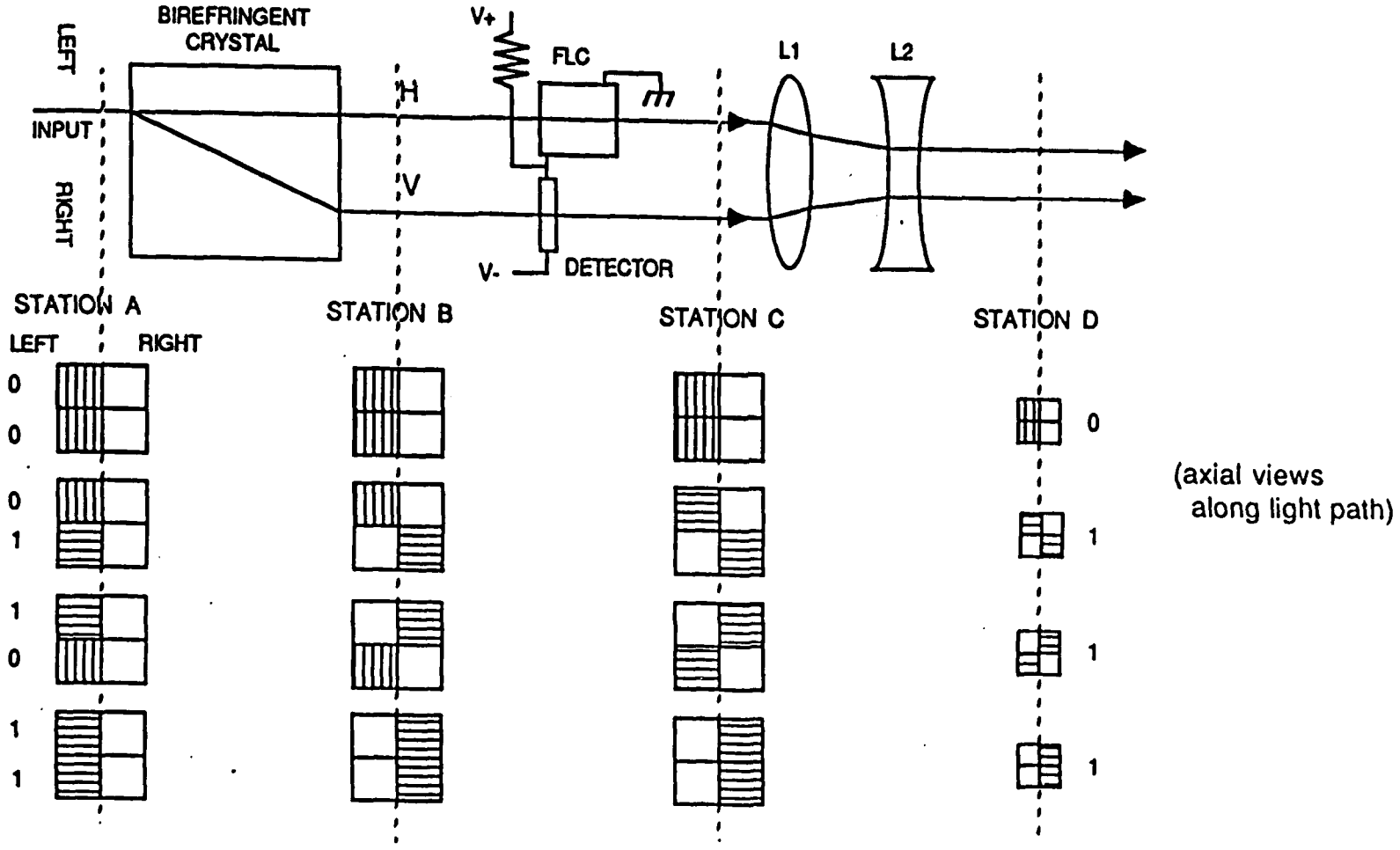

FIGURE 7, POLARIZED LIGHT LOGIC GATE

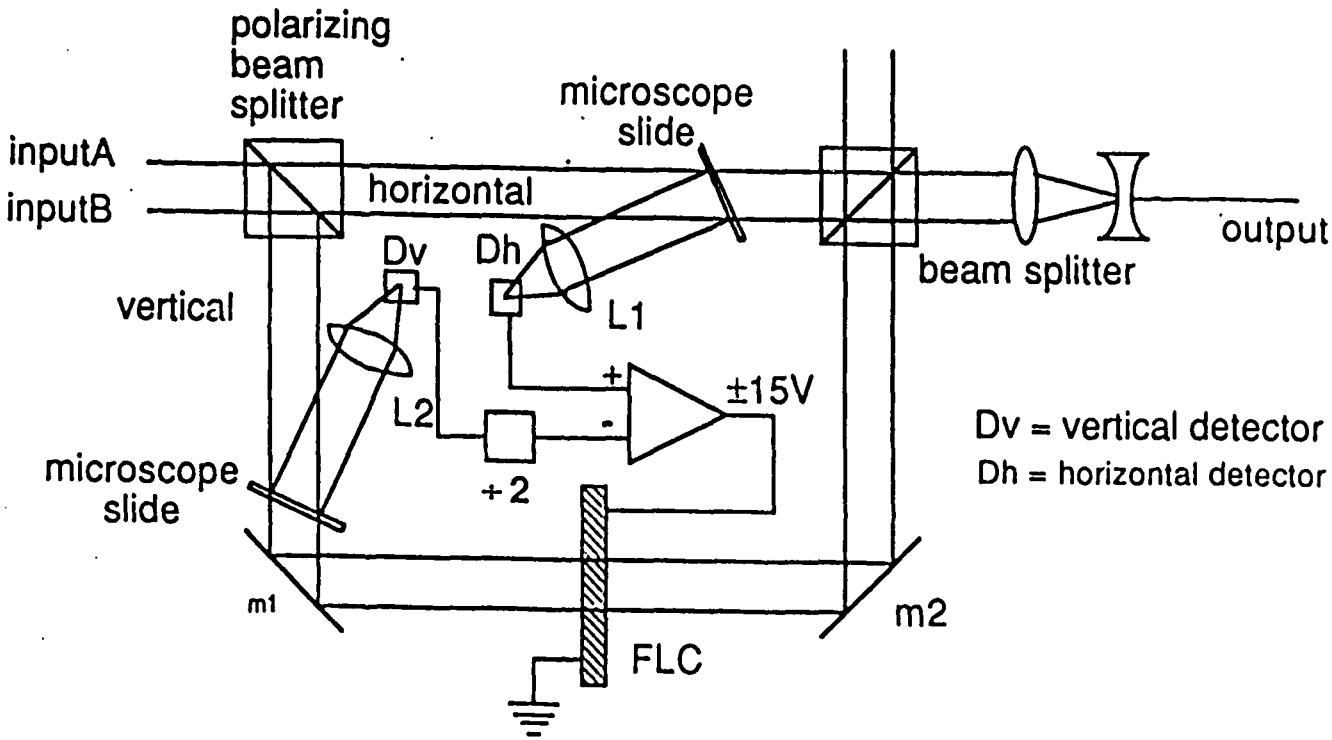

FIGURE 8, PROTOTYPE LOGIC GATE

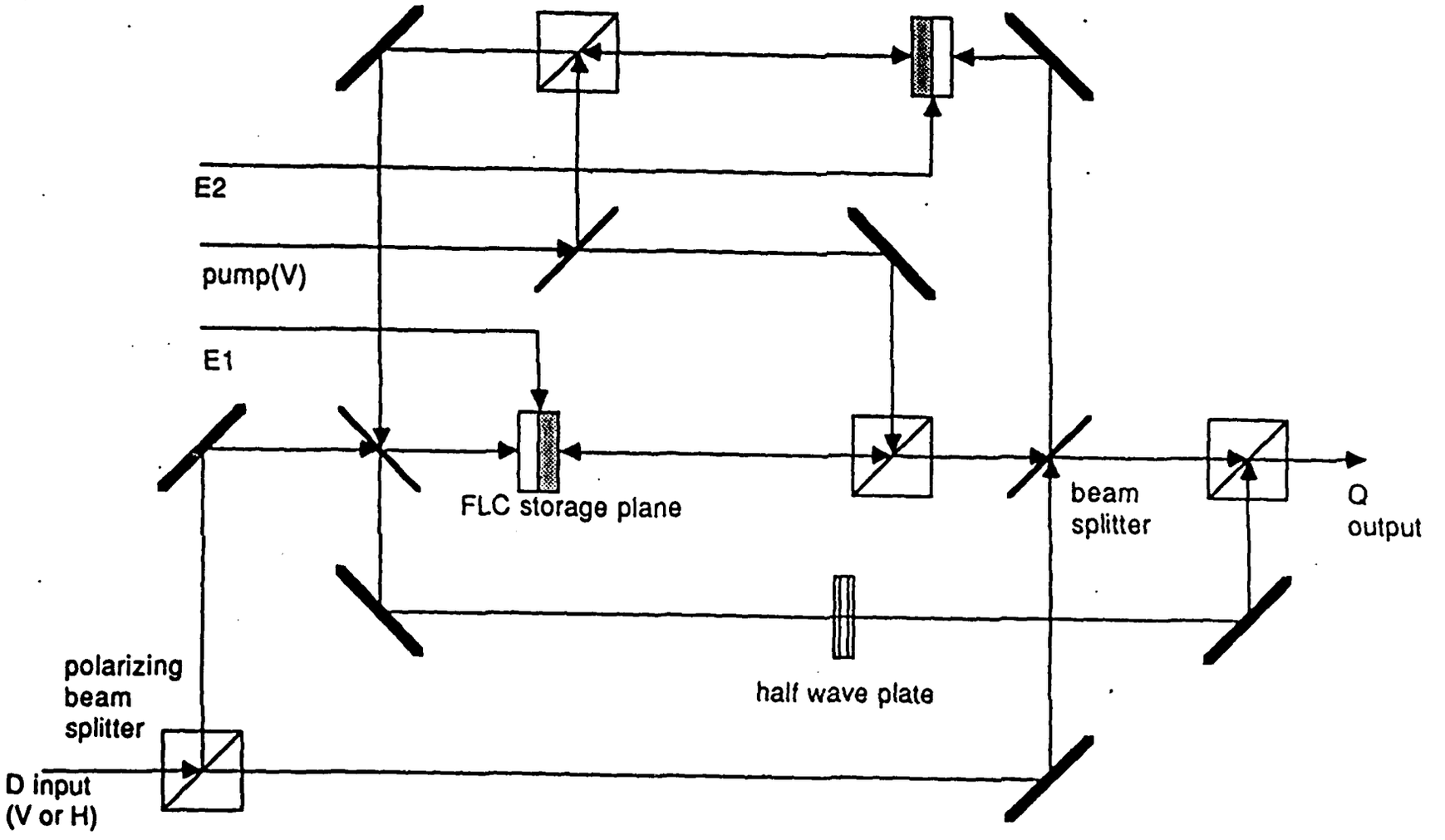

FIGURE 9, FLC MEMORY FRAME

APPENDIX I

A REASONING PROBLEM

(From Winston, Artificial Intelligence, p. 177)

1) If the animal has hair then it is a mammal.
2) If the animal gives milk then it is a mammal.
3) If the animal has feathers then it is a bird.
4) If the animal flies and it lays eggs then it is a bird.
5) If the animal is a mammal and it eats meat then it is a carnivore.
6) If the animal is a mammal and it has pointed teeth and it has claws and its eyes point forward then it is a carnivore.
7) If the animal is a mammal and it has hoofs then it is an ungulate.
8) If the animal is a mammal and it chews cud then it is an ungulate and it is even-toed.
9) If the animal is a carnivore and it has a tawny color and it has dark spots then it is a cheetah.
10) If the animal is a carnivore and it has a tawny color and it has black stripes then it is a tiger.
11) If the animal is an ungulate and it has long legs and it has a long neck and it has a tawny color and it has dark spots then it is a giraffe.
12) If the animal is an ungulate and it has a white color and it has black stripes then it is a zebra.
13) If the animal is a bird and it does not fly and it has long legs and it has a long neck and it is black and white then it is an ostrich.
14) If the animal is a bird and it does not fly and it swims and it is black and white then it is a penguin.
15) If the animal is a bird and it flies then it is an albatross.

PROVE:

- If the animal has hair and it has hoofs and its color is white and it has black stripes then it is a zebra.
- If the animal has a tawny color and it has dark spots and it gives milk and it chews cud and it has a long neck and it has long legs, then it is a giraffe.

These rules are translated into a Lisp input as:

```
(hairy implies mammal)
(lactates implies mammal)
(feathers implies bird)
((flies and (lays eggs)) implies bird)
((mammal and (eats meat)) implies carnivore)
((mammal and (pointed teeth) and claws and (looks forward)) implies carnivore)
((mammal and hoofs) implies ungulate)
((mammal and (chews cud)) implies (ungulate and (even toed))
((carnivore and tawny and (dark spots)) implies cheetah)
((carnivore and tawny and (black stripes)) implies tiger)
((ungulate and (long legs) and (long neck) and tawny and (dark spots)) implies giraffe)
((Ungulate and white and (black stripes)) implies zebra)
((bird and (not flies) and (long legs) and (long neck) and (black and white)) implies
ostrich)
((bird and (not flies) and swims and (black and white)) implies penguin)
((bird and flies) implies albatross)
*
((hairy and hoofs and white and (black stripes)) implies zebra
```

In clause form this becomes:

```
1' + 2
3' + 2
4' + 5
6' + 7' + 5
2' + 8' + 9
2' + 10' + 11' + 12' + 9
2' + 13' + 14
2' + 15' + 14
2' + 15' + 16
9' + 17' + 18' + 19
9' + 17' + 20' + 21
14' + 22' + 23' + 17' + 18' + 24
14' + 25' + 20' + 26
5' + 6 + 22' + 23' + 27' + 28
5' + 6 + 29' + 27' + 30
5' + 6' + 31
*
1
13
25
20
26'
```